BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan Dan Koordinasi

Kerja magang selama 3,5 bulan dilaksanakan di Universitas Tokyo Denki yang berlokasi di 2-1200 MuzaiGakuendai, Inzai, Chiba 270-1382, Jepang berkedudukan sebagai *programmer*. Dalam kerja magang ini, dirancang dan dibuat sebuah *project* bernama "Tomolivi" (Together More in Living Room) yang berkaitan dengan komunikasi antar manusia menggunakan berbagai sensor untuk menghubungkan keluarga yang terpisah, bersama dengan 3 rekan kerja lainnya. Kerja magang diawasi oleh Prof. Naoki Mukawa selaku kepala laboratorium dan dikoordinasi oleh Murakami Miyu selaku ketua *project*, juga dibimbing oleh Saito Natsuki selaku penanggung jawab.

Secara keseluruhan, sistem program ini merupakan sistem komunikasi baru yang memanfaatkan perpaduan program Kinect selaku pendeteksi keberadaan dan posisi manusia, Arduino selaku sensor cahaya dan tekanan serta XBee sebagai koneksi *wireless* antar Arduino. Sistem ini nantinya dapat digunakan lagi sebagai dasar untuk mengembangkan *project-project* di kemudian hari.

3.2 Tugas yang Dilakukan

Selama periode magang selama 3,5 bulan, dibuat program pendeteksi posisi beberapa manusia dengan menggunakan kinect, koneksi arduino, hubungan sensor dan *server*, serta *graphical user interface* yang akan ditampilkan. Didapatkan tanggung jawab yang sama dengan Stevanus Kevin

Santana yang melakukan praktek kerja magang di tempat yang sama sehingga dari tanggung jawab yang diberikan dibagi menjadi dua sehingga tanggung jawab yang dilakukan adalah sebagai berikut.

1. Mendeteksi posisi *user* melalui Kinect.
2. Mengaktifkan *multiple tracking* Kinect.
3. Mengkonversi data *input* dari Kinect menjadi data yang cocok untuk ditampilkan.
4. Menghubungkan sensor-sensor ke *server* melalui komputer.
5. Mengirim data yang dibutuhkan ke *server*.

Untuk pengambilan data dari *server*, desain *graphical user interface*, dan pemrosesan data dari *server*, serta menampilkannya secara *real time* akan dikerjakan oleh Stevanus Kevin Santana.

Prinsip kerja sistem yang akan dibangun adalah kinect bertugas untuk membaca posisi *user*, sensor cahaya yang siap menerima input cahaya, dan sensor tekanan yang memberikan informasi tentang duduk/tidaknya *user*. Informasi-informasi yang telah diambil akan diproses dan dikirimkan ke *user* lawan bicara melalui *server*. Program Kinect dibuat menggunakan bahasa pemrograman C++, Arduino dan XBee menggunakan bahasa Java/Processing, dan pemrograman *server* menggunakan PHP.

Pada pembuatan sistem yang ditugaskan, pembuatan sistem bisa dikategorikan sebagai pengembangan dari detail metode *System Development Life Cycle (SDLC)* dalam implementasinya. Oleh karena itu, untuk mengeneralisasi

tugas-tugas yang diberikan selama pembuatan sistem ini, pembuatan mengacu pada fase-fase dalam System *Development Life Cycle (SDLC).*

Karena sistem ini terbagi menjadi 2 bagian besar, yaitu Kinect dan Arduino, maka *System Development Life Cycle (SDLC)* ini dibagi menjadi 2 bagian.

Tabel 3.1 *Timeline* Sistem

No	Kegiatan	September				Oktober				November				December	
		1	2	3	4	1	2	3	4	1	2	3	4	1	2
1	Studi literatur														
2	*Requirement analysis*														
3	*Design*														
4	*Development*														
5	*Integration and Test*														
6	*Implementation*														
7	*Operation and Maintenance*														
8	Penulisan laporan dan dokumentasi														

3.2.1 Requirement Analysis

Selama periode magang ini, didapatkan tugas untuk membuat sebuah sistem yang menghubungkan kinect dan sensor-sensor lainnya yang menggunakan Arduino ke sebuah *server* yang berperan sebagai *database* pada sistem. Semua

data yang telah terbaca disimpan dalam bentuk *text file* yang berekstensi .txt pada *server*.

Untuk sensor Kinect, sistem memerlukan kemampuan untuk mendeteksi posisi manusia pada sebuah ruangan. Kinect tersebut akan diletakkan di ujung ruangan, dan akan menerima informasi posisi dari seluruh orang yang ada dalam ruangan tersebut. Untuk dapat melakukan hal ini, pertama-tama program harus dapat mendeteksi keberadaan manusia secara plural. Selain itu, program harus dapat mempertahankan hasil *scan* secara *continuous*. Dalam hal ini, yang dimaksudkan adalah program dapat membedakan posisi manusia yang satu dengan yang lain. Dengan semua informasi ini, posisi setiap manusia yang ada pada ruangan dapat terdeteksi. Batas deteksi hanya sampai 6 orang karena 6 orang adalah batas maksimum yang bisa dideteksi oleh kinect. Sistem diberikan jeda waktu untuk mengirim data ke *server* agar performa sistem tidaklah terlalu berat. Program kinect ini dibangun dengan bahasa C++.

Requirement kedua yang diminta adalah sistem diharuskan dapat mendeteksi keberadaan cahaya dan tekanan pada ruangan selama sistem menyala. Untuk memenuhi *requirement* ini, tentunya membutuhkan sensor yang dapat mendeteksi intensitas cahaya dan sensor yang dapat mendeteksi tekanan fisik. Dalam hal memenuhi *requirement* ini, diputuskan untuk menggunakan Arduino sebagai sensor cahaya dan tekanan.

Arduino adalah semacam pengendali mikro *single-board* yang bersifat *open source*, diturunkan dari *wiring platform*, dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang. Berbagai kelebihan Arduino,

seperti harga yang murah, penggunaan yang sederhana, kompabilitas yang tinggi dengan berbagai macam sensor, serta *open source* membuat tim pemgembang memutuskan untuk menggunakan arduino sebagai sensor yang akan digunakan pada sistem.

Dikarenakan ada lebih dari satu sensor pada Arduino, tim pengembang memutuskan untuk menggunakan satu Arduino untuk masing-masing sensor lalu mennyambungkan semua sensor tersebut ke sebuah Arduino yang disebut *coordinator.*

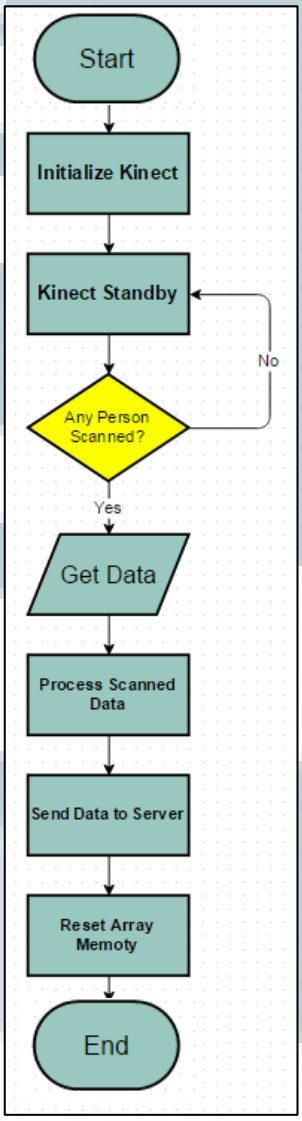
Selain dari hasil analisis penggunaan *hardware-hardware* yang telah ditetapkan dan hasil analisis *user requirement* secara teknis, ditarik beberapa kesimpulan yang perlu diterapkan dalam sistem, yaitu sebagai berikut.

a. Semua perangkat *hardware* haruslah terhubung ke *server* untuk mengirimkan data.
b. Masing-masing program akan memproses data terlebih dahulu sebelum mengirim atau menerima data.
c. Untuk menghindari *deadlock*, jeda waktu pengambilan data dan pengiriman data dibuat sama persis satu dengan yang lain.
d. Jeda waktu juga diberikan untuk meringankan beban komputer.
e. Koneksi antar Arduino bersifat *wireless*.

3.2.2 Design

Pada tahap perancangan, dilakukan pembuatan dan analisis algoritma yang dibutuhkan dan dipakai untuk rancangan sistem ini. Untuk membuat rancangan suatu sistem, tentunya dibutuhkan gambaran dan rancangan teknis tentang alur kerja dari sistem yang dibuat.

A. Kinect



Gambar 3.1 *Flow Chart* Kinect

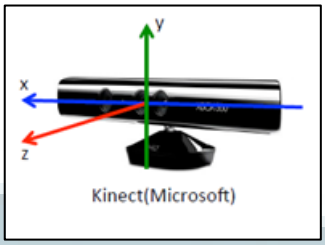
Flow Chart yang telah dibuat menjelaskan seluruh aktivitas Kinect mulai dari saat dinyalakan sampai berhasil menyimpan data pada *server* kegiatan ini terus berlangsung sampai Kinect dimatikan. Adapun kegiatan Kinect yang harus dikembangkan adalah Kinect harus bisa mendeteksi presensi banyak orang sekaligus (sampai 6 orang), mencatat posisi masing-masing orang yang terdeteksi lalu mengirimkan data ke *server* tiap jeda waktu 10 detik, dan me-*reset* kembali *array memory* yang digunakan untuk menampung data. Berikut adalah penjelasan dari masing-masing aktivitas yang terdapat pada gambar 3.1.

1. *Kinect Standby*

   Pada tahap awal ini, program kinect telah dinyalakan dan telah terhubung dengan *server*. Kinect telah siap menerima *input* posisi setiap *user* yang berada pada jarak pandang Kinect dan telah terdeteksi oleh kinect. Jika tidak ada *user* yang terdeteksi oleh Kinect, maka Kinect akan tetap pada posisi *standby*. Namun jika ada *user* yang terdeteksi oleh Kinect, program akan mengambil informasi posisi masing-masing *user* yang terdeteksi.

2. *Get Data*

   Pada tahap ini *user* telah terdeteksi oleh kinect. Program kemudian mulai mengambil koordinat posisi *user* terhadap Kinect. Format posisi *user* yang diambil Kinect disimpan dalam koordinat (X,Y,Z) dimana posisi X,Y, dan Z dapat dilihat secara jelas pada gambar 3.2.



Gambar 3.2 Posisi X, Y, dan Z pada Kinect
(Sumber: *en.wikipedia.org/wiki/Kinect*)

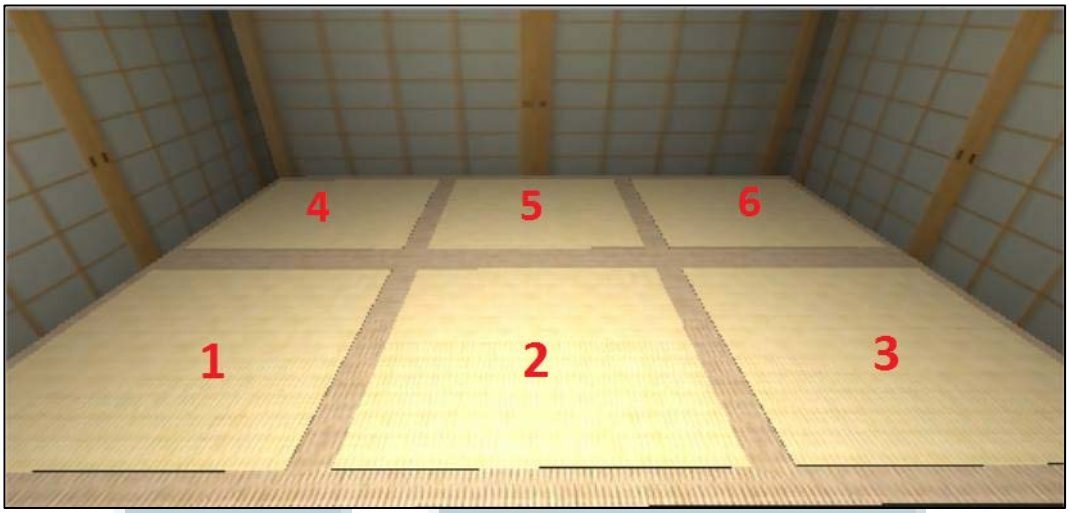
Kinect mengambil data secara *multithread* sehingga koordinat harus ditampung secara bersamaan dan bisa diketahui *thread* mana yang memiliki koordinat yang dimaksud. Gambaran potongan data yang didapatkan oleh Kinect dapat dilihat pada tabel 3.2.

Tabel 3.2 Ilustrasi Data yang Didapat oleh Kinect

Thread No.	X(Pixel)	Y(Pixel)	Z(mm)	Time
2	551	247	1223	18:36:33
2	576	297	1152	18:36:33
3	462	107	2257	18:36:33
2	483	166	1188	18:36:34

3. *Process Scanned Data*

Pada tahap ini, Kinect akan memproses semua data yang telah diambil menjadi data yang lebih mudah untuk diproses pada *layer Graphical User Interface*. Tim pengembang sepakat untuk membagi ruangan yang dideteksi oleh Kinect dibagi menjadi 6 bagian agar mempermudah saat menentukan posisi *user* saat menampilkannya di *graphical user interface*. Ilustrasi pembagian denah ruangan dapat dilihat pada gambar 3.3.



Gambar 3.3 Pembagian Denah Ruangan

Pada tahap ini, Kinect akan mengubah data yang telah diambil menjadi data yang sesuai dengan ilustrasi di atas. Ilustrasi potongan data yang telah diproses dapat dilihat pada tabel 3.3.

Tabel 3.3 Ilustrasi Data setelah Diproses

Thread 1 Position	Thread 2 Position	Thread 3 Position	Thread 4 Position	Thread 5 Position	Thread 6 Position	Time
-	3	6	-	-	-	18:36:40
-	3	-	-	-	-	18:36:50
-	2	-	-	-	-	18:37:00

Data tersebut nantinya akan dikirim ke *server* untuk ditampung.

4. *Send Data to the Server*

Pada tahap ini, program Kinect akan mengirimkan data yang telah ditampung selama 10 detik ke *server* menggunakan *File Transfer Protocol* (FTP). Data akan dikirim dalam bentuk *text file* berekstensi .txt ke *server*.

5. *Reset Array Memory*

Setelah data dikirimkan ke *server*, *array memory* yang digunakan untuk menampung data selama 10 detik akan di-*reset*. Tujuan *reset* ini adalah mencegah ambiguitas data apabila jumlah orang yang terdeteksi berkurang.

Tanggung jawab dilakukan atas segala aktivitas yang dikerjakan Kinect, kecuali saat inisialisasi. Inisialisasi program Kinect dikerjakan oleh *programmer* lain. Tanggung jawab yang dilakukan dapat dijabarkan menjadi sebagai berikut.

1. Mendeteksi posisi *user* melalui Kinect

   Bagian pendeteksian ini adalah *looping* yang selalu terjadi selama program Kinect berjalan. Proses ini adalah yang menentukan apakah Kinect akan lanjut ke *step* berikutnya atau tidak.

2. Mengaktifkan *multiple tracking* Kinect

   Posisi *user* yang harus bisa di-*track* oleh Kinect tidak hanya satu posisi saja. Kinect harus bisa mendeteksi lebih dari satu orang. Oleh karena itu, kemampuan *multiple tracking* yang memungkinkan Kinect untuk mendeteksi lebih dari satu orang saat mendeteksi posisi *user* harus diaktifkan.

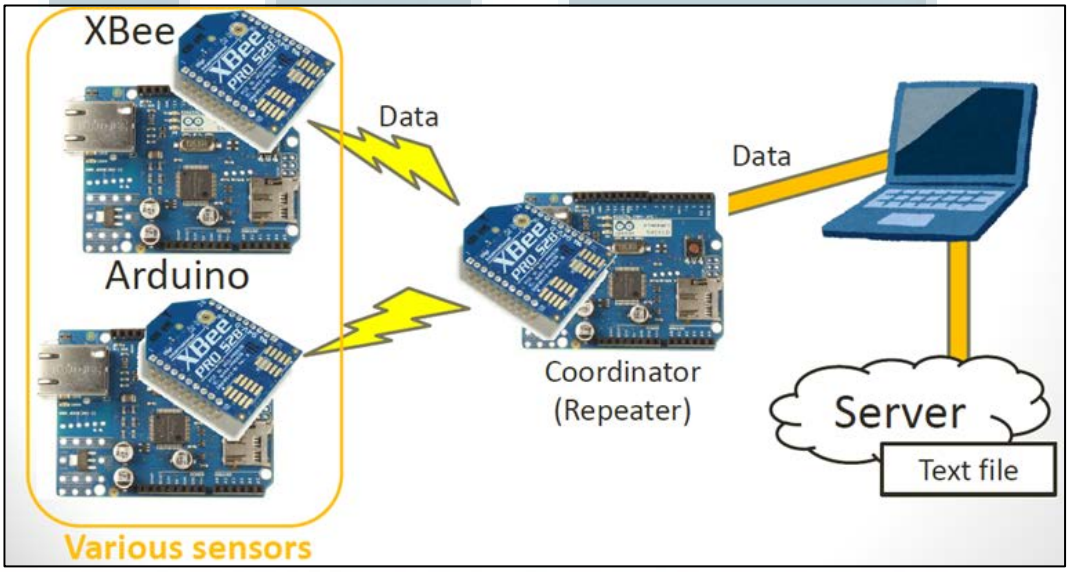
3. Mengkonversi data input dari Kinect menjadi data yang cocok untuk ditampilkan

   Data yang diberikan Kinect sebelum diproses tidaklah cocok untuk menjadi patokan sebagai data yang ditampilkan ke *interface* karena struktur

datanya berantakan dan tidak memiliki interval waktu yang baik untuk penyajian data. Interval waktu yang terlalu cepat akan memperberat sistem saat transmisi data ke *server*.

B. Arduino dan XBEE

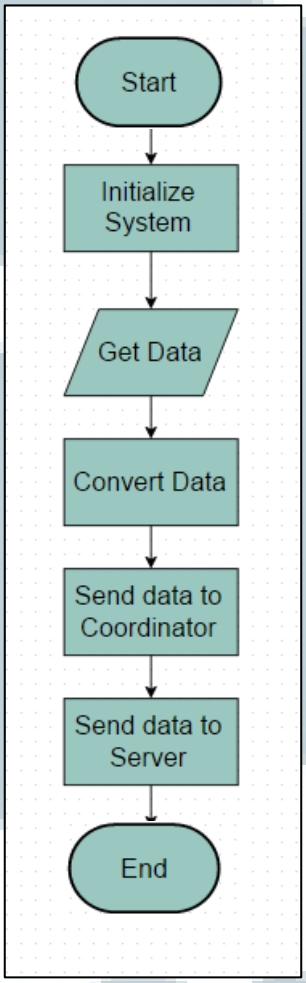
Arduino dan XBee berperan sebagai sensor pendeteksi intensitas cahaya dan tekanan pada keseluruhan sistem "tomolivi" ini. Gambaran sistem Arduino dapat dilihat pada gambar 3.4.



Gambar 3.4 Gambaran Sistem Arduino
(Sumber: *forum.arduino.cc*)

Pada pengembangan sistem ini, digunakan dua buah sensor pada Arduino, yang mendeteksi keberadaan cahaya dan keberadaan tekanan pada ruangan. Kedua sensor tersebut dihubungkan ke sebuah sensor *coordinator* secara *wireless*. Sensor *coordinator* ini berperan sebagai penghubung antar berbagai sensor dan komputer. Hubungan antar sensor terkoneksi melalui XBee, yang berperan sebagai *wireless adapter* untuk Arduino. *Coordinator* terhubung ke komputer

dengan menggunakan kabel LAN. Komputer ini yang nantinya akan mengoperasikan *coordinator* dan mengirim data ke *server*. *Flow chart* Arduino dapat dilihat pada gambar 3.5.



Gambar 3.5 *Flow Chart* Arduino

Flow Chart pada gambar 3.5 menggambarkan aktivitas Arduino secara keseluruhan, mulai dari dinyalakan sampai dimatikan. Adapun kegiatan Arduino yang harus dikembangkan adalah menjembatani koneksi antara Arduino yang berperan sebagai *coordinator* dengan *server*. Berikut adalah penjelasan dari *flow chart.*

1. *Initialize System*

Pada tahap ini, semua sensor telah menyala dan terhubung antar satu dengan yang lain. Sensor akan mengambil data sesuai dengan perannya masing-masing apabila sistem telah *stand by.*

2. *Get Data*

Pada tahap ini, sensor akan mengambil data yang dibutuhkan sesuai dengan perannya masing-masing. Data yang diterima sensor masih dalam bentuk *default* sehingga tidak cocok untuk penampilan data dan harus dikonversi terlebih dahulu.

3. *Convert Data*

Pada tahap ini, data yang diterima sensor akan dikonversi sehingga lebih cocok dan mudah dimengerti ke penampilan *user interface*. Semua sensor mengirim data dalam bentuk Boolean.

4. *Send Data to the Coordinator*

Pada tahap ini, data yang telah dikonversi masing-masing sensor akan dikirimkan ke *coordinator* untuk dikumpulkan dan dikirim ke *server* secara bersamaan. Di tahap ini juga, *coordinator* membedakan data antar sensor.

5. *Send Data to the Server*

Data yang telah diterima *coordinator* dikirim ke *server* secara bersamaan. Program *coordinator* akan mengirimkan data langsung ke PHP pada *server.* Selanjutnya, *server* akan men-*generate text file* berekstensi .txt secara otomatis sesuai dengan program yang dibuat dengan PHP.

Tanggung jawab dilakukan atas koneksi Arduino *coordinator* dengan *server*, dan program PHP yang terdapat pada *server*. Untuk inisialisasi, pengambilan data, dan koneksi antar Arduino dikerjakan oleh *programmer* lain. Tahapan dan cara-cara yang digunakan selama pengerjaan proyek berpatokan dengan desain yang telah dijelaskan di atas.

Kedua sistem tersebut nantinya akan dijalankan bersamaan, tetapi independen satu dengan yang lain. Kerusakan pada satu sensor tidak akan mempengaruhi kinerja sensor yang lain. Data yang dikirimkan ke *server* oleh kedua sistem tersebut akan disatukan oleh program *user interface* yang dibuat oleh *programmer* lain, Stevanus Kevin Santana.

3.2.3 Development

Selama proses pengembangan dari sistem ini, ada beberapa perangkat keras dan perangkat lunak yang akan digunakan. Berikut ini adalah penjabaran perangkat keras yang digunakan.

1. *Laptop* Sony® Vaio VPCCB16FG sebagai perangkat yang digunakan selama pengembangan dengan spesifikasi umum sebagai berikut.
    a. *Processor*: Intel® Core™ i5-2410M (4 cores, 3MB cache, 2.30GHz with Turbo Boost up to 2,90GHz)
    b. VGA: AMD Radeon HD6630M 1GB + INTEL® HD Graphics 3000
    c. RAM: 4GB DDR3
    d. *Harddisk* dengan kapasitas 750GB

2. Microsoft® Kinect Version 1 yang digunakan sebagai pendeteksi posisi.
3. Arduino sebagai perangkat keras yang digunakan untuk membaca sensor cahaya dan tekanan serta *coordinator*.
4. XBee PRO S2B sebagai koneksi *wireless* antar Arduino.

Beberapa perangkat lunak yang digunakan selama tahap pengembangan sistem antara lain sebagai berikut.

1. Sistem operasi Windows 7
2. Visual C++ 2010 Express yang digunakan sebagai *environment* untuk membuat aplikasi kinect
3. *Software* arduino yang digunakan untuk melakukan *programming* arduino
4. *Software* Java/Processing yang digunakan untuk melakukan *programming* koneksi Arduino ke *server*
5. *Software* XCTU untuk melakukan *setup* koneksi antar Arduino
6. FFFTP sebagai program *File Transfer Protocol* untuk mengecek data pada *server*
7. Google Chrome *web browser* yang digunakan untuk mencari informasi perihal pengembangan sistem
8. Microsoft Word 2010 untuk membuat dokumentasi dan laporan kerja magang
9. Microsoft Power Point 2010 untuk presentasi proyek
10. Notepad sebagai *environment* untuk PHP *programming*

3.2.4 Integration and Test

Selama pengembangan sistem, ada beberapa pengujian yang dilakukan terhadap mekanisme kerja dari bagian-bagian sistem yang diuraikan sebagai berikut.

1. *Testing* Kinect

   Pengecekan dilakukan dengan cara mencoba langsung menjadi *user*, mencoba untuk berpindah ke berbagai posisi (masih dalam jarak pandang kinect) pada ruangan sehingga dapat menganalisis respon Kinect terhadap perubahan posisi *user*. Tim juga mencoba *multiple tracking* dengan menambah jumlah *user* yang dideteksi oleh Kinect, serta respon Kinect atas segala kemungkinan yang terjadi saat *tracking.* Perubahan data pada *server* juga selalu dimonitori agar meyakinkan program Kinect berjalan dengan baik dan benar. Respon Kinect yang lambat sebelum berhasil mendeteksi orang seringkali mempersulit tahap *testing* ini.

2. *Testing* Sensor Arduino

   Testing sensor pada Arduino ini cukup mudah dilakukan karena respon Arduino cukup cepat sehingga mempermudah pengambilan data. Data yang diberikan Arduino sangatlah sesuai dengan harapan tim. Namun, diberikan *delay* 10 detik antar *scan* agar sistem tidak terlalu terbebani. Sama seperti Kinect, pengecekan dilakukan dengan memonitori perubahan data pada *server* agar meyakinkan bahwa Arduino bekerja dengan baik.

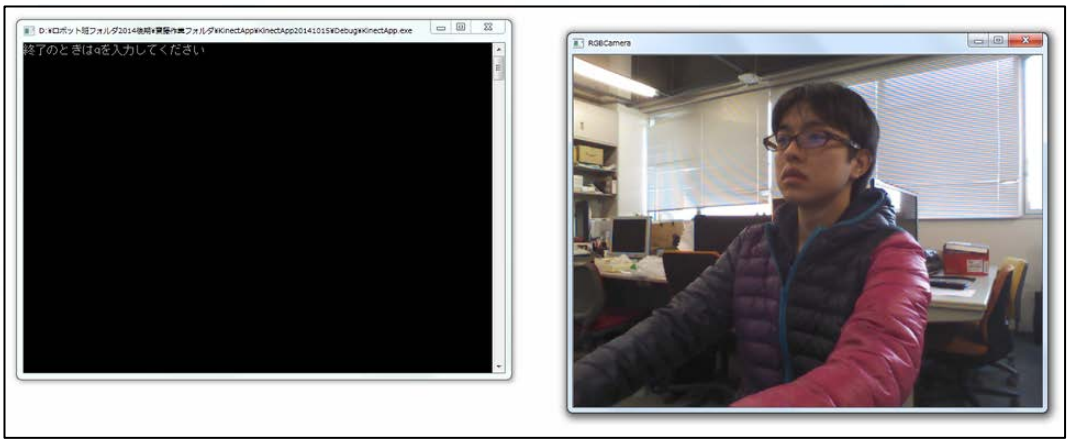
Testing dilakukan oleh beberapa pihak. *Testing* seringkali dilakukan pada saat *development* agar mempermudah dan mempercepat proses perbaikan dan

penyempurnaan. *Testing* juga diakukan oleh profesor pembimbing dan *project leader*, dan dilakukan uji coba pada beberapa *user* yang dimintai tolong untuk memberikan saran. Dari hasil *testing* yang dilakukan oleh beberapa *user*, sempat terjadi beberapa kali perbaikan terkait dengan cara kerja dan penyajian data.

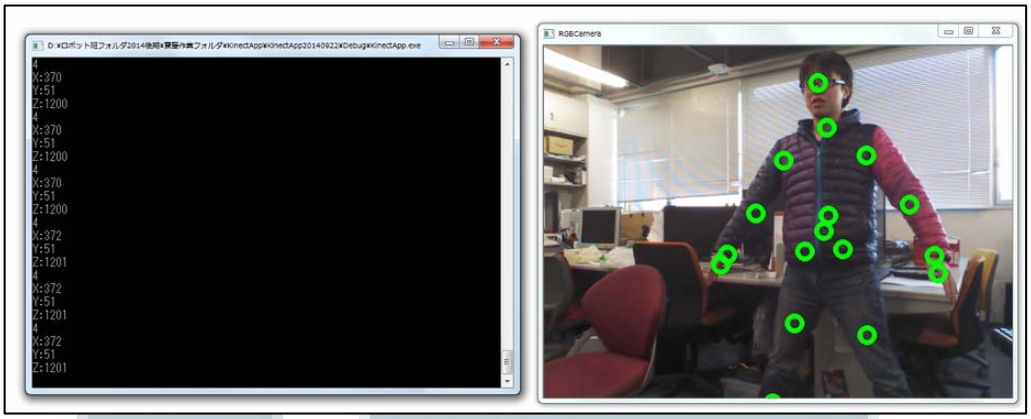
3.2.5 Implementation

Pada tahap implementasi ini, digabungkan berbagai sub program yang telah dibuat sebelumnya. Seperti yang telah dijelaskan pada tahap *design*, nantinya program kinect dan program sensor Arduino akan dijalankan bersamaan dengan program Interface yang telah dibuat oleh rekan satu tim, Stevanus Kevin Santana.

Saat sistem akan dijalankan, Kinect dan semua sensor yang ada harus dalam keadaan *stand by.* Masing-masing program dapat berjalan sendiri. Program hanya akan mempengaruhi perubahan data pada *server* secara *real time*. Program *interface* juga hanya merespon dari data yang terdapat pada *server* dan akan mengubah tampilan secara *real time* apabila terjadi perubahan pada data yang diakibatkan oleh program Kinect atau program Arduino yang berjalan.



Gambar 3.6 Sistem Kinect saat *Stand By*



Gambar 3.7 Sistem Kinect saat *Tracking*

User yang badannya terlihat oleh Kinect akan dideteksi oleh Kinect, kemudian program Kinect akan menampilkan koordinat dari posisi *user* terhadap Kinect. Koordinat ditampilkan secara *real time* pada bagian konsol program C++ sehingga dilihat secara langsung respon Kinect terhadap perubahan posisi dan data yang terjadi. Koordinat yang berhasil dideteksi oleh Kinect akan diproses menjadi bagian ruangan seperti yang terlihat pada gambar 3.3. Koordinat X dan Z pada Kinect akan menentukan pada ruangan bagian mana *user* berada.

Kinect bersifat *multithreading* saat mendeteksi keberadaan manusia. Hanya terdapat 6 *thread* program pada Kinect yang berfungsi untuk mendeteksi posisi manusia. Hal ini menyebabkan sistem hanya dapat mendeteksi sampai 6 orang saja. *Thread-thread* inilah yang bekerja saat Kinect berhasil mendeteksi keberadaan orang. Masing-masing *thread* akan mendapatkan informasi tentang koordinat dari titik pinggang orang yang terdeteksi.

Informasi keberadaan *user* ini akan ditampung pada *memory array* yang berlaku sebagai *thread* tersebut. Dibuat array 1 dimensi untuk menampung data dari setiap *thread* dalam bentuk koordinat sehingga data yang tersimpan dapat

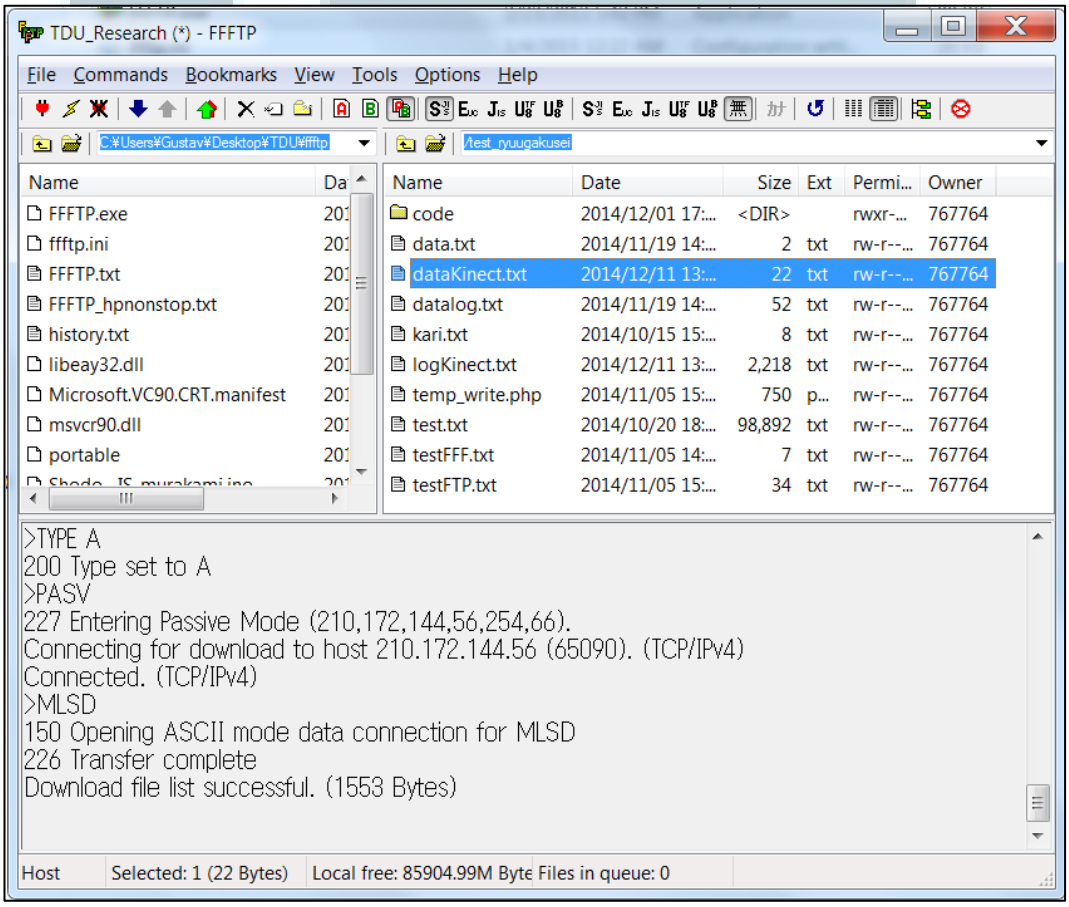
terlihat seperti pada tabel 3.2. Data ini kemudian akan disimpan dalam bentuk *text file* berekstensi .txt dan dikirimkan ke *server* melalui *File Transfer Protocol* (FTP) yang menjadi satu dengan program Kinect. Gambar 3.8 menunjukkan tampilan potongan program yang berperan sebagai FTP.

```
void FTP(void)
    {
    CkFtp2 ftp;
    bool success;
    success = ftp.UnlockComponent("FTPConnect");
    if (success != true) {
        printf("%s¥n",ftp.lastErrorText());
        return;
    }
    ftp.put_Hostname("ftp.doc.whitesnow.jp");
    ftp.put_Username("whitesnow.jp-doc");
    ftp.put_Password("a18H4w8P");
    //  Connect and login to the FTP server.
    success = ftp.Connect();
    if (success != true) {
        printf("%s¥n",ftp.lastErrorText());
        return;
    }
    //  Change to the remote directory where the file will be uploaded.
    success = ftp.ChangeRemoteDir("test_ryuugakusei");
    if (success != true) {
        printf("%s¥n",ftp.lastErrorText());
        return;
    }
    //  Upload a file.
    const char * localFilename;
    localFilename = "data.txt";
    const char * remoteFilename;
    remoteFilename = "dataKinect.txt";
    success = ftp.PutFile(localFilename,remoteFilename);
    if (success != true) {
        printf("%s¥n",ftp.lastErrorText());
        return;
    }
    localFilename = "log.txt";
    remoteFilename = "logKinect.txt";
    success = ftp.PutFile(localFilename,remoteFilename);
    if (success != true) {
        printf("%s¥n",ftp.lastErrorText());
        return;
    }
    ftp.Disconnect();
    printf("File Uploaded!¥n");
    }
```

Gambar 3.8 Potongan Program FTP

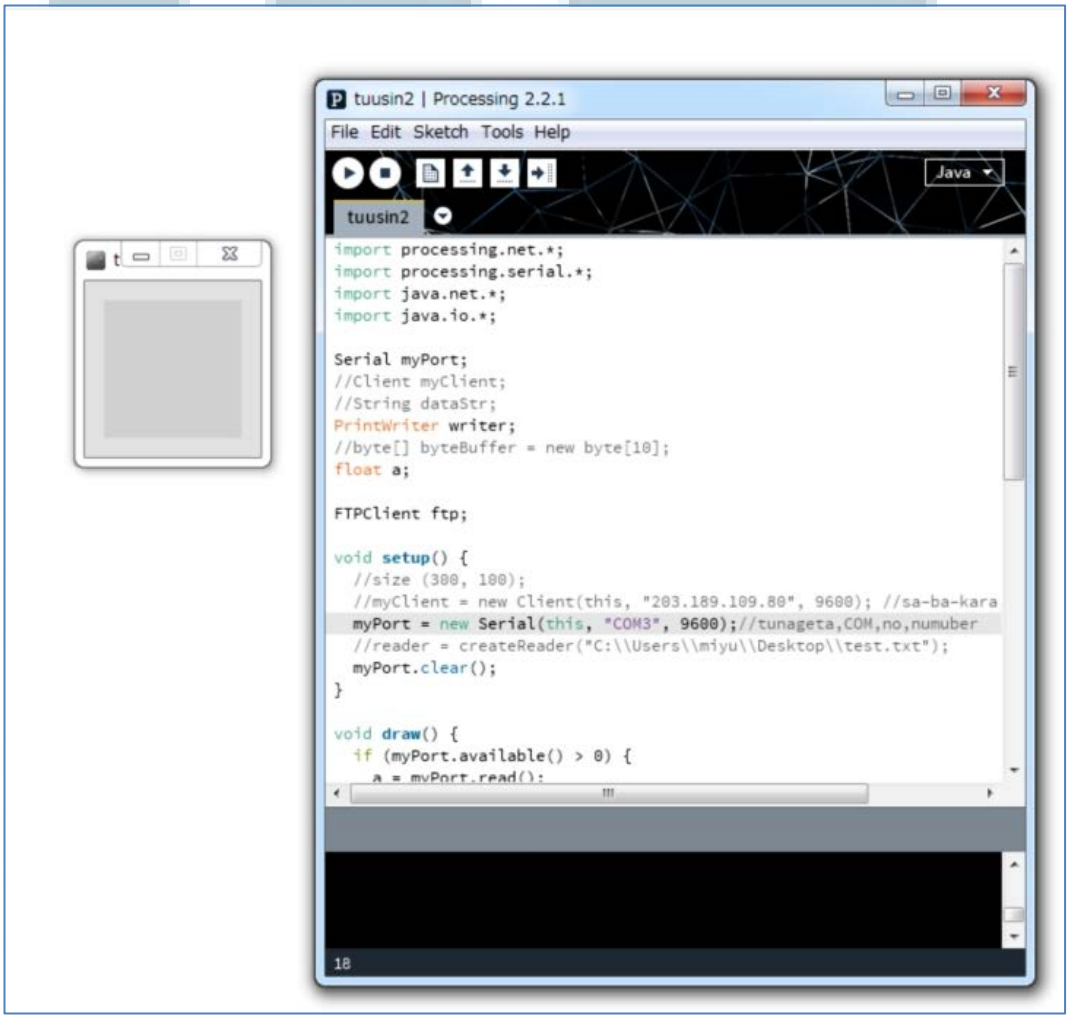
Program tersebut akan mengirimkan data yang telah direkam oleh Kinect selama 10 detik. Data dikirimkan dengan interval 10 detik. *Delay* 10 detik ini diberikan untuk menjaga performa setiap *thread* pada Kinect. Apabila data dikirimkan setiap saat, kinerja *thread* pada Kinect akan sangat terganggu sehingga tidak dapat menyajikan data dengan baik.

Perubahan data dapat dimonitori dengan menggunakan *software* FFFTP. FFFTP adalah *software* aplikasi yang memungkinkan transfer *file* antara komputer *user* dengan *server*. Dengan *software* ini, tim memonitori segala perubahan data yang terjadi baik pada data Kinect maupun data Arduino. Contoh tampilan dari program FFFTP dapat dilihat pada gambar 3.9.



Gambar 3.9 Tampilan *Software* FFFTP

Sebuah koneksi dibuat untuk menjembatani antara *coordinator* dan *server*. Koneksi tersebut dibuat dengan menggunakan Java/Proccessing sebagai *environment* pembuatannya. Program pada sensor lainnya dibuat dengan menggunakan bahasa pemrograman Arduino, tetapi untuk pemrograman pada *coordinator* dibuat menggunakan Java/Processing. Hal ini dilakukan karena tim ingin memonitori kerja *coordinator* dengan Java/Processing. Selain itu, tahap pengembangan juga menjadi lebih mudah karena Java/Processing dapat menambah *library*.



Gambar 3.10 Tampilan Program *Coordinator*

Berbeda dengan Kinect, *coordinator* Arduino mengirim langsung data ke program PHP pada *server* sehingga program PHP itulah yang bekerja dan membuat *log file* yang dibutuhkan dari hasil kerja *coordinator*. Kinect masih menggunakan FTP dikarenakan tim pengembang ingin melakukan *socket programming* untuk menambah jumlah Kinect pada penelitian selanjutnya.

3.2.6 Operation and Maintenance

Tahap *operation* diserahkan kepada ketua proyek dan penanggung jawab laboratorium.

Tahap *maintenance* sudah dua kali dilakukan, sekali pada program Kinect dan sekali pada program Arduino. *Maintenance* pertama dilakukan karena data yang diberikan Kinect tidak mudah diproses pada program *interface*, juga tidak cocok sebagai patokan data sebagai data yang disimpan pada *server* karena informasi yang diberikan tidak lengkap sehingga dibuat metode program Kinect seperti yang dijelaskan pada laporan ini. Sementara, pada *maintenance* kedua dilakukan perbaikan pada metode pengiriman data Arduino, dari FTP menjadi PHP *server programming* seperti yang telah dijelaskan pada subbab 3.2.2 bagian B.

3.2.7 Penulisan Dokumentasi dan Laporan Magang

Penulisan dokumentasi dan laporan magang dilakukan selama proses pengembangan sistem berlangsung. Dibuat catatan apa saja yang dilakukan dan pencapaian setiap minggu. Selama proses kerja magang berlangsung, laporan

magang juga turut dibuat untuk diserahkan ke pihak Universitas Tokyo Denki bagian laboratorium interaksi. Laporan yang telah dibuat dan diserahkan ke pihak Universitas Tokyo Denki dalam format *powerpoint presentation* dengan bahasa Inggris yang berisikan latar belakang, metode yang digunakan, *flow* program, penjelasan sistem, dan penjelasan algoritma yang digunakan pada program.

Pembuatan laporan dimaksudkan agar nantinya sistem yang telah dibuat dapat digunakan maupun dikembangkan oleh anggota laboratorium. Laporan dipresentasikan untuk melaporkan perkembangan dan hasil akhir sistem.

Sebagai gambaran yang lebih jelas, detail realisasi kerja magang yang dilaksanakan selama bekerja sebagai *programmer* di Universitas Tokyo Denki dapat dilihat pada tabel 3.4.

Tabel 3.4 Realisasi Kerja Magang

Minggu	Kegiatan
1	- Pengenalan tempat kerja dan anggota - Pembahasan dan pemilihan *project*
2	- Penginstallan visual c++ 2010 express - Pembahasan *project* lebih lanjut
3	- Pengenalan dan pembelajaran koding *skeletal tracking* menggunakan Kinect - Studi Literatur mendalam tentang Kinect

Tabel 3.4 Realisasi Kerja Magang (Lanjutan)

Minggu	Kegiatan
4	- Mendapatkan koordinat user - Mempelajari *multiple tracking*
5	- Mengaktifkan *multiple tracking* - Diskusi konsep pengambilan data
6	- Mengambil data hasil *tracking* - *Upload* data ke *server*
7	- Pengenalan dan pembelajaran *programming* arduino - Penginstallan program *Arduino*
8	- Membantu membuat rangkaian program pendeteksi cahaya - Penginstallan program *Processing*
9	- Penulisan metode pengiriman data dengan FTP pada Arduino - Mempersiapkan presentasi untuk melaporkan perkembangan proyek
10	- Mengubah metode FTP menjadi PHP programming
11	- Evaluasi ulang sistem Kinect - Diskusi tentang perubahan data flow kinect
12	- Maintenance program Kinect, merubah data yang dihasilkan kinect
13	- Evaluasi akhir sistem keseluruhan - Mempersiapkan presentasi akhir
14	- Presentasi akhir

3.3 Kendala dan Solusi

3.3.1 Kendala yang Ditemukan

Berikut kendala-kendala yang ditemukan selama proses kerja magang berlangsung.

1. Kesulitan dalam memahami permintaan maupun ekspektasi tugas yang harus dicapai, serta interaksi antar anggota laboratorium karena hambatan bahasa.
2. Kurang pengetahuan terkait pemrograman kinect serta sensor-sensor sehingga mengalami kesulitan dalam mengembangkan sistem
3. Kinerja yang terlalu sesuai jadwal, tidak *flexible* saat dapat mempercepat pekerjaan. Sebaliknya, akan kesulitan jika pekerjaan terlambat.
4. Tidak adanya *job description* yang jelas antara sesama mahasiswa magang terkait tugas dan tanggung jawab dikarenakan kurangnya persiapan penerimaan program kerja magang.

3.3.2 Solusi Atas Kendala

Solusi yang sekiranya dapat mengatasi kendala di atas adalah sebagai berikut.

1. Menggunakan teknologi pengalih bahasa untuk berkomunikasi serta mendalami pembelajaran bahasa Jepang untuk memudahkan komunikasi.
2. Mendalami pengetahuan terkait pemograman kinect dan sensor-sensor melalui *forum* internet dan modul-modul yang ada.

3. Berinisiatif mempercepat pekerjaan dengan langsung bertanya kepada *project leader* untuk tugas berikutnya.
4. Banyak berkomunikasi dengan rekan laboratorium agar pembicaraan saat *project* dilaksanakan dapat disampaikan dengan baik.